Interoperability Architectures

Mr. Jonathan Searle, Mr. John Brennan
Simulation and Synthetic Environment Laboratory (SSEL)
Defence Academy of the United Kingdom
Cranfield University
UK

J.R.Searle@cranfield.ac.uk / j.m.brennan@cranfield.ac.uk

INTRODUCTION

The first paper in this series addressed the general concepts of interoperability within the modelling and simulation (M&S) realm. The next stage, within the context of the lecture series as a whole, is to examine interoperability from an integration perspective. To accomplish this, one must look at the manner in which simulations (and their components) can be *bolted* together. Furthermore, to ensure a complete assessment is conducted one must consider more than just the technical aspects of interoperability and integration; it is critical that one gives due attention to the organisational, cultural and economic aspects of integration as well. As such, one can state that the main challenges of interoperability and integration within the M&S realm are composed of a mix of engineering, technical and conceptual issues.

The primary aim of this paper is to consider the **integration** and **interoperability** of M&S with particular emphasis on **federated** and **distributed** simulations.

To address this aim it is important for one to begin with a basic understanding of the significant terms within the statement, in the context of M&S. First of all, the term interoperability is meant to imply the ability of a given system to exchange data or information, and perform its required functions in concert with other distinctly separate systems. The term integration refers to the induction or amalgamation of M&S systems into existing programmes or studies such as training and acquisition. Therefore, in *simulation speak*, interoperability relates to the creation of *federations* whereas integration relates to the application of the federations. Finally, one must also explore the issues of composability and architectures – two fundamental elements underpinning interoperability and integration within the synthetic environment realm.

Prior to launching into the main element of this paper, it is worth taking the time to briefly explore the concepts of federated and distributed. These two terms are related, in some cases, but they do not necessarily imply mutual dependence within the M&S realm. Federated systems are systems that contribute to a whole while each element maintains self internal management – a federation is an organisation formed by merging several elements. The term distributed can have two meanings within the context of this paper. First, the term distributed computing generally refers to a collection of independent computers each performing partial elements of a greater task, which appears to any user as a single coherent system. Within the M&S realm, distributed has a similar meaning except that it normally also carries the implication of physical distribution (e.g. geographically disparate). The main reasons for employing distributed simulation are reliability, scalability, connectivity and extensibility (or composability). Essentially, it is important for one to understand that the term federation (or federated) does not imply distributed, or vice versa.

Searle, J.; Brennan, J. (2006) Interoperability Architectures. In *Integration of Modelling and Simulation* (pp. 4-1 – 4-8). Educational Notes RTO-EN-MSG-043, Paper 4. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **01 SEP 2006**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Interoperability Architectures**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Simulation and Synthetic Environment Laboratory (SSEL) Defence Academy of the United Kingdom Cranfield University UK**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001943., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**39**	

COMPOSABILITY

Composability as a concept is not new. The idea of building a system from individual (proven) components has been common within engineering for many years (e.g. the automotive industry). Similarly, the software engineering world adopted this concept, particularly with the advent of object oriented programming techniques. If one examines the implementation of a simulation, we can see that composability exists in many forms at a variety of levels. Take for instance, flight simulation. One can purchase a commercially available, PC-based flight simulator *game* (e.g. Microsoft's Flight Simulator 2004) and see that it is in fact a composition of many individual elements, or components, of software code. A component of code handles the flight dynamics of the aircraft, while other components handle elements such as weather modelling, terrain modelling and computer generated air traffic. Not too dissimilar, is the typical civil aviation full flight simulator. These complex systems are constructed in a conceptually similar manner except that rather than simply a collection of software components, they are a collection of components, each comprised of hardware and software. Nonetheless, individual components will still perform specific functions as mentioned above, and the various hardware components will be networked to provide the user with the illusion that he is operating a single, coherent aircraft.

One can take this concept of composability to even higher levels wherein multiple platform simulators/simulations are linked together to provide a common mission space. This permits small team training, such as that necessary for flight elements or packages in the military fast jet arena. Extend yet again by bringing in land, maritime and C4ISR[1] assets, and one can create (compose) a joint simulation space with varied combinations of assets. Figure 1 attempts to capture this concept.

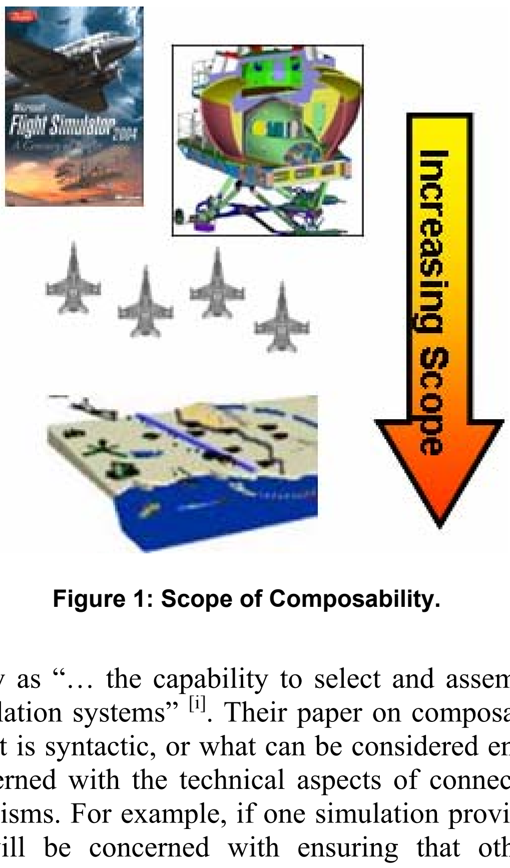


Figure 1: Scope of Composability.

Petty et al define composability as "… the capability to select and assemble simulation components in various combinations into simulation systems" [i]. Their paper on composability goes on to describe two types of composability. The first is syntactic, or what can be considered engineering level composability. At this level one is most concerned with the technical aspects of connecting components through data interfaces or invocation mechanisms. For example, if one simulation provides position as an output, then this level of composability will be concerned with ensuring that other networked simulations or components can receive and interpret the position data packet when it is sent. The second type of composability mentioned is semantic, or modelling composability. At this level, consistent assumptions

[1] C4ISR – Command, Control, Communications, Computers, Intelligence, Surveillance, Reconnaissance.

and domain validity become important. That is, one is more concerned with whether or not the composition of two or more models is actually meaningful. For example, if one model issues an order to attack, do the other models that might receive this order *understand* what is meant by the attack order.

Essentially, the desire to assemble individual models or simulations as components of a larger system is certainly not a new concept. Nonetheless, one must be sure to address composability at both the technical level and the semantic (or modelling) level. For if there is no logical reason or meaning behind attempts to network individual components, no amount of assurance at the technical level will guarantee successful execution of the final system.

ARCHITECTURES FOR DISTRIBUTED SIMULATION

With the concept of building higher order systems by connecting multiple components having been examined, it is appropriate now to begin exploring the ways in which one can envision organising the components to facilitate a logical assembly. There are different conceptual approaches for categorising the elements of a distributed simulation (or federation). Two possible methods are to distribute by function and distribute by platform.

When distributing by function, one can begin by categorising the component models or simulations by military environments. For example, in a joint simulation system one could have a land battle component, a maritime battle component and an air battle component. Similarly, there would be components for such things as weather modelling, terrain modelling, ballistics modelling and sensor modelling. Figure 2 depicts this concept, which schematically represents the construct on which the Joint Training Confederation (JTC) is built. [ii]

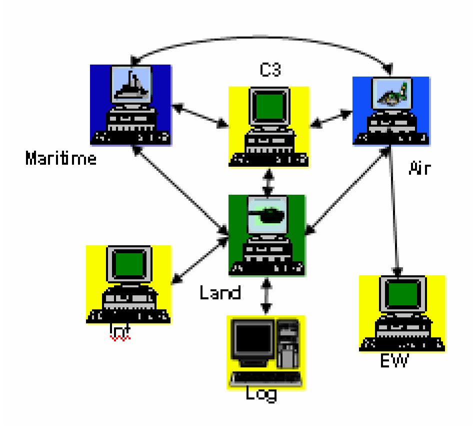


Figure 2: Distributed Simulation by Function.

Another approach to distributed simulation, one that is likely more familiar, is by platform. This architecture is essentially a peer-to-peer distribution of equals (i.e. of similar components). Each station (or node) generates platform-level entities and all simulations/entities communicate with each other. One example of this approach is the IEEE 1278 standard known as Distributed Interactive Simulation (DIS). The DIS standard is based on a message passing protocol wherein ground truth state data is passed via protocol data units (PDUs) such that all participants potentially know all ground truth. This approach has benefits in the training domain but it is not very flexible beyond that.

Architectures for distributed simulation follow the conceptual examples above. First, the distribute by function concept is implemented as a network of functional nodes (see Figure 3). This architecture is optimal for applications other than training, such as the work and research done by industry or other like organisations. The primary advantage is that this architecture is relatively more easily reconfigured to address a variety of studies or applications; as such, notwithstanding that it can be more complex, it is deemed more composable and has greater potential for reuse.

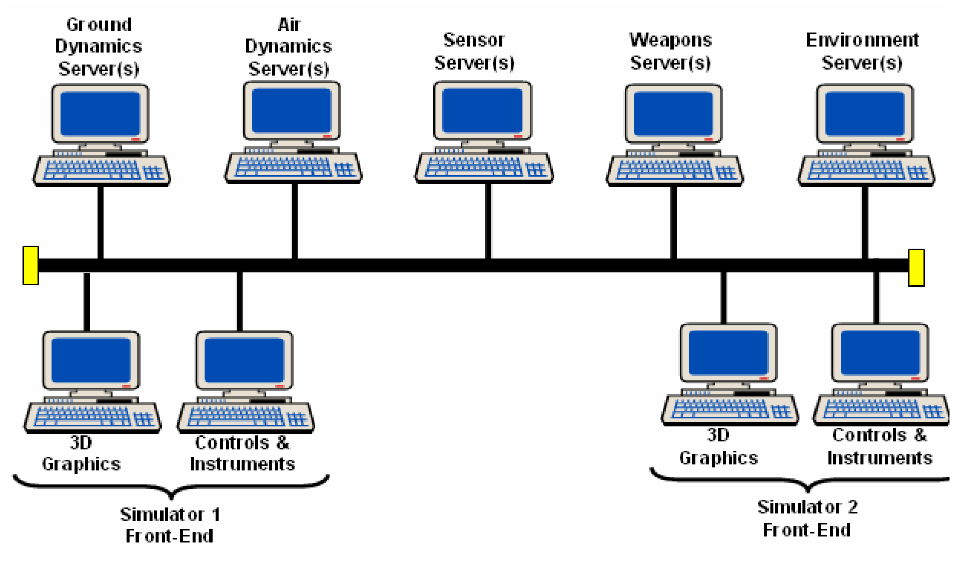


Figure 3: Functionally Distributed Architecture.

The distribute by platform concept is represented by a discrete, monolithic architecture (see Figure 4). Essentially a network of single computers or simulators, this architecture is good for training but not very flexible or scalable (i.e. one cannot go on adding more and more nodes *ad infinitum*). The systems that participate in this style of architecture are fairly homogenous and any modifications made to one participant may necessitate source code changes to all federates.

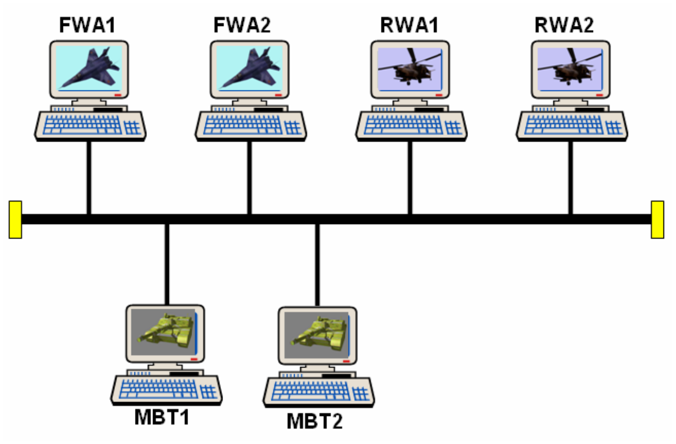


Figure 4: Discrete (Platform) Distributed Architecture.

INTEROPERABILITY AND INTEGRATION

When looking to build a federation, it is clearly a good starting point if each of the federates involved seemed to represent the same sort of things in a similar way. So, as an example, consider two hypothetical battle group level, ground combat simulations, which both represent individual vehicle platforms:

- An older legacy model, developed for stand-alone analysis use; and
- A newer model, developed from the outset as a training component.

One might ask the question, "How interoperable might they be?" Is it possible for the older simulation to be modified to make it interoperable with the newer system (i.e. to achieve reuse of the old model)? Furthermore, if this were possible, would it be sensible to do so?

The more traditional combat model was written in an older programming language, using traditional coding methods and styles. It was designed as a closed system, and therefore owns, controls and updates each and every entity within the environment on its own. Being a discrete event simulation[2], the speed of execution for the older system is complexity dependent. If executing in real-time, the simulation executes a small *battle slice* as quickly as possible (**much** faster than real-time), then simply pauses (sits idle) to let real-time catch up. At a lower level, the advances in processing actually occur in *bursts* (see Figure 5).

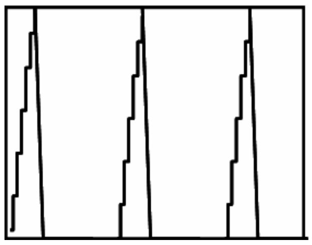


Figure 5: CPU Usage & Event Processing – Older System.

On the other hand, the newer model, like most current day systems, was designed and programmed in an object oriented manner. Although still event driven, it was written as proper real-time software, incorporating cyclic *callbacks* with very small time steps (see Figure 6). Consequently, the system executes consistently with relatively smooth processing and time advance.

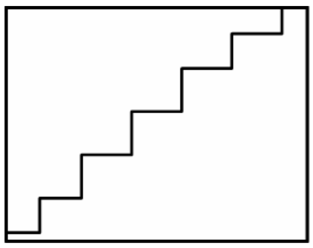


Figure 6: CPU Usage & Event Processing – Newer System.

By way of example, the preceding two paragraphs begin to draw out some of the differences one can encounter when attempting to make two or more simulation systems interoperate. The following table identifies additional instances of the more common interoperability issues one might come upon.

[2] A discrete event simulation is one in which the state variables are updated periodically, in distinct steps. Discrete event simulations are typically time driven or event driven.

Typical Simulation Interoperability Issues		
Item	**Legacy Simulation**	**Newer Simulation**
Terrain	Matrix of cells	Often polygon based (but does not have to be)
Vehicle Orientation	Location & general orientation	Vehicle x, y, z and hdg, pitch, roll (i.e. 6 DoF)
Vehicle Space Occupation	Abstract (i.e. point mass)	Occupies physical space (i.e. bounding box)
Movement	Updated in discrete time slices	Updated more continuously (smoothly)
Engagement	Probability table (Hit & Kill); model determines all effects	Firing unit calculates ballistics & impact point; target calculates effects

The two simulation systems described are relatively similar in that they are both used to model ground combat; however, one can clearly see that there are issues which must be addressed. Furthermore, what if the simulations were fundamentally different in application; one would expect that the issues would be even more pronounced.

To be successful, interoperability requires due consideration to a range of issues. The initial problem is to establish how simulation state data will be shared when hindered by the physics, characteristics and topology of networks. Even with modern networked systems, one cannot underestimate the impact and costs of these issues. Compounded on this are the issues of security and intellectual property rights; these are particularly crucial in the military domain, when dealing across borders or with industry. But beyond this, there should also be concern regarding the meaningful interpretation and use of the data to be exchanged by each simulation and tool involved; technically two systems may be able to exchange data and even then understand it, but having done so, does it actually make sense? Essentially, in addition to considering the implementation level one must consider the conceptual level of interoperability that underpins the activity.

When considering the conceptual level, architecting interoperability must account for natural groupings of models and associated levels of interoperability. The typical hierarchical pyramid of models (see Figure 7) helps visualise natural groupings and the associated application areas of the wide expanse of models available today.

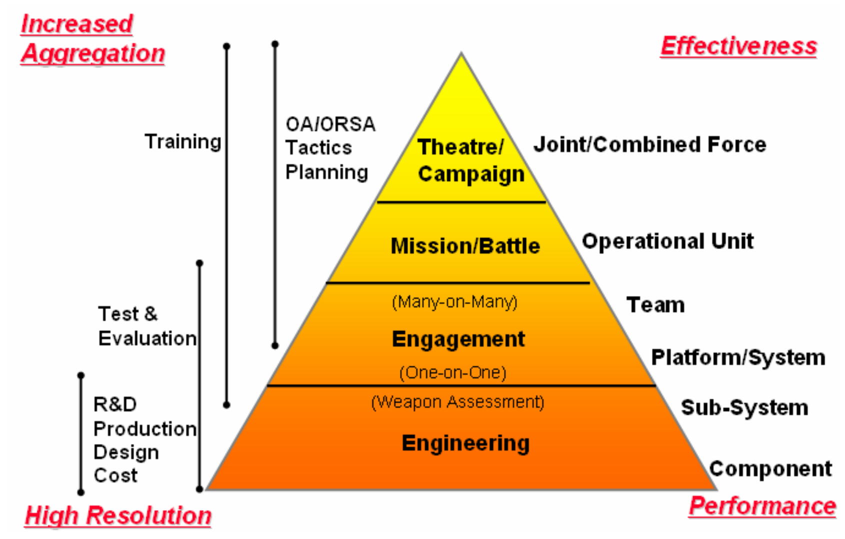


Figure 7: Hierarchical Pyramid of Models.

A second (or indeed alternative) way of establishing potential opportunities for interoperability is to use the approach that is now being termed *communities of interest* (COI). Essentially another way of classifying high level disciplines, COIs can represent groups of people or organisations that are focused on areas such as research and development, training or command and control (engineering, platform and aggregate foci as depicted in Figure 8).

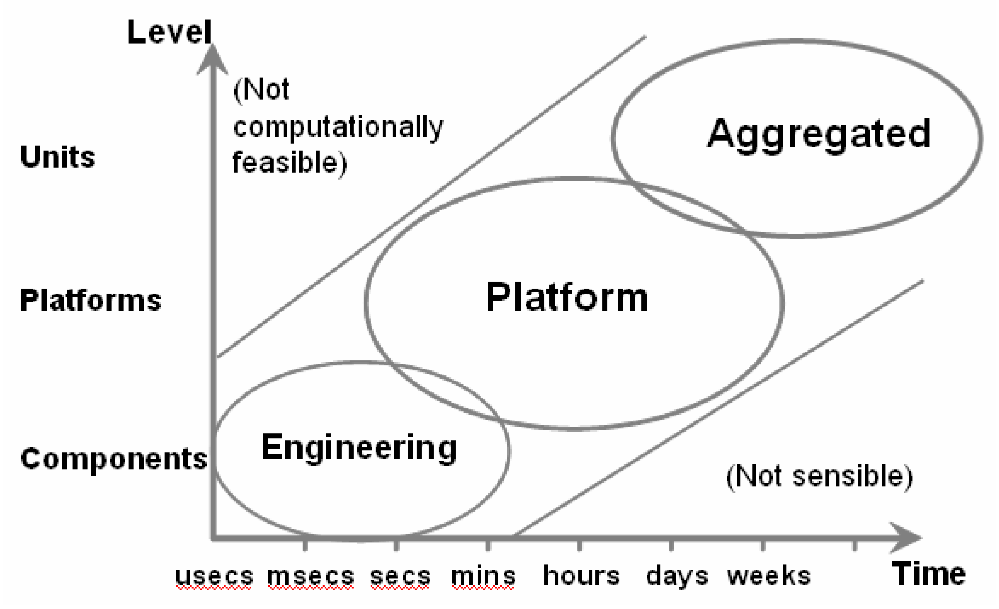


Figure 8: Communities of Interest Concept (Graham Shanks, AMS, 2005).

One final element worth mentioning is the effort put forth by the European Cooperation for Long-term In Defence (EUCLID) research and Technology Program (RTP) 11.13 – the Synthetic Environment Development and Exploitation Process (SEDEP)[iii]. Dubbed *Realising the Potential of networked Simulation in Europe* the SEDEP efforts aimed to provide tools and processes to mitigate obstacles to effective use of synthetic environments (SEs) within Europe. In short, the SEDEP program recommended that, in order to enhance collaboration and reuse in a complex environment, one would need to provide processes and tools interfaced to repositories, and shared data environments and knowledge bases. Conclusions such as these tend to emphasise the fact that the use of SEs is problematic because of integration as much as because of interoperability.

SUMMARY

Networked and distributed simulation federations have come a long way since the development of SimNet in the early 1980s. The scope has increased dramatically necessitating much more complex architectures. The increased complexity has encouraged the practice of composability and reuse to help ease and facilitate the construct of complex federations. The architectures must account for not only the technical and engineering aspects but also the practical and conceptual characteristics of a federation. No single architecture can provide an all encompassing solution for all applications – each requirement must be approached independently in the first instance – then one can look to composability and reuse as means of providing efficiency.

REFERENCES

[i] Taken from *Overview of a Theory of Composability*, Petty, Weisel & Mielke.

[ii] See http://alsp.ie.org/alsp/joint/index.html for more information.

[iii] See www.euclid1113.com for more details.

NATO RTO Lecture Series
MSG-043
Integration of Modelling & Simulation

Interoperability Architectures

Jonathan Searle & John Brennan

Defence Academy – Cranfield University

Simulation and Synthetic Environment Laboratory

Engineering Systems Department

Outline

- Aim:
  - To consider the integration and interoperability of modelling and simulation, with particular emphasis to federated and distributed simulation.

- This means examining:
  - Composability
  - Architectures
  - Interoperability
  - Integration

M&S Integration & Interoperability

- Interoperability
  - With other M&S
  - With other things
    - Operational CIS
- Integration
  - Within training programmes
  - Within studies
  - Within acquisition programmes

Federations

Applications of federations

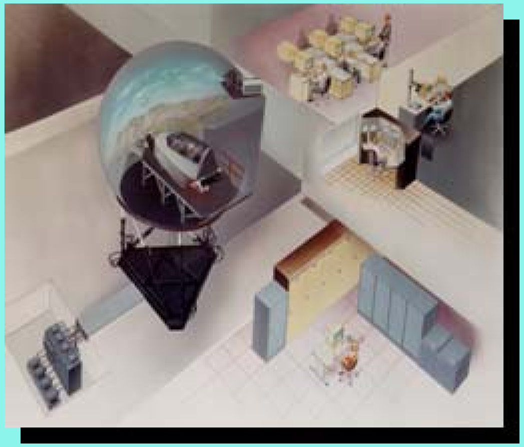
- The main challenges in Interoperability are usually a mix of the engineering and technical and the conceptual.
- The main challenges in Integration are organisational, cultural and economic.

Why Distribute M&S - 1

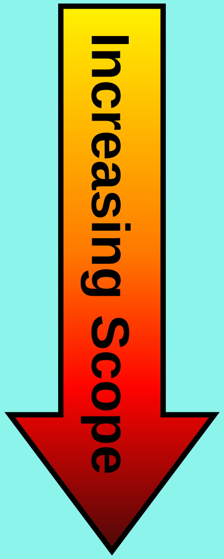
- ***Reliability***:
  - Avoiding single points of failure (eg using centralised servers).
- ***Scalability***:
  - To enable multi-user access and participation.
  - To share processing loads across systems (distributed computing).
- ***Connectivity***:
  - For live interaction between remote sites.
  - To drive (‘stimulate’) other resources from simulations.
  - To share data, information and knowledge.
- ***Extensibility & Composability***:
  - To add capability or functionality to existing M&S apps.
  - To build complex systems from multiple components.

System Building

- Building from components is not a new concept.
- Software Engineering advances permit just that.



- If we examine the implementation of ‘a’ simulation:
  - Single box system : will have multiple, ‘connected’ software components.
  - Complex simulator: has multiples of such boxes, networked together, representing one platform.
  - Mission simulation: connects multiple platform simulators, perhaps also with C4ISR systems.
  - Joint simulation: connects even more varied combinations

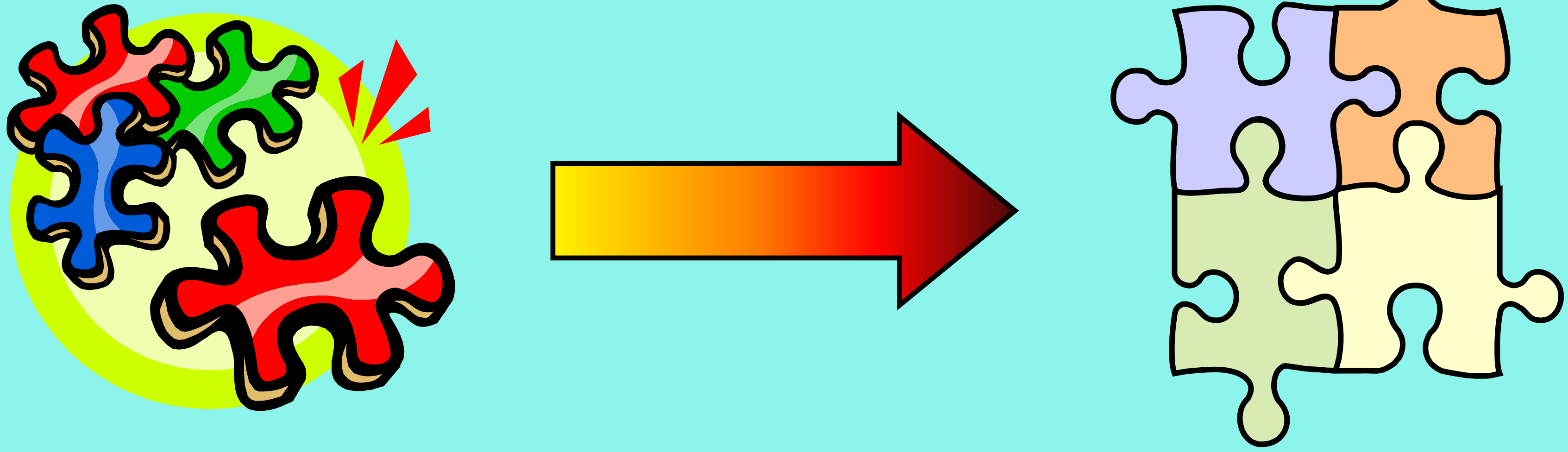


- Advances in M&S (and its application areas) are moving us along this road.
- Paralleling enterprise software development in the business field.

Composability

- Composability is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements.

[Overview of a Theory of Composability : Petty, Weisel & Mielke]



Why Distribute M&S - 2

An example from one application domain, examining the connection and interoperation of UK training systems:

Why?

- Operations more complex
- More flexible task groupings & Effects Based Operations
- NEC & exploitation and expansion of the digitized battlespace
- Increased emphasis on the joint nature of command
- Rapid and graduated response

Hence:

- More flexible, less bespoke and stove-piped training capabilities
- More training delivered with/through operational systems
- Better training management and use of assets
- Reduced overheads for collective training

[Fawkes, UK MoD DAES, Def Sim & Trg Conf 2004]

Composability

Two types of composability:

- Syntactic (engineering) composability
  - Implementing composable components
  - Concerned with component connectability

    e.g., data interfaces, invocation mechanisms
  - Can the components be combined?

- Semantic (modelling) composability
  - Ensuring composable models
  - Concerned with model compatibility

    e.g., domains of validity, consistent assumptions
  - Is the composition of models meaningful?

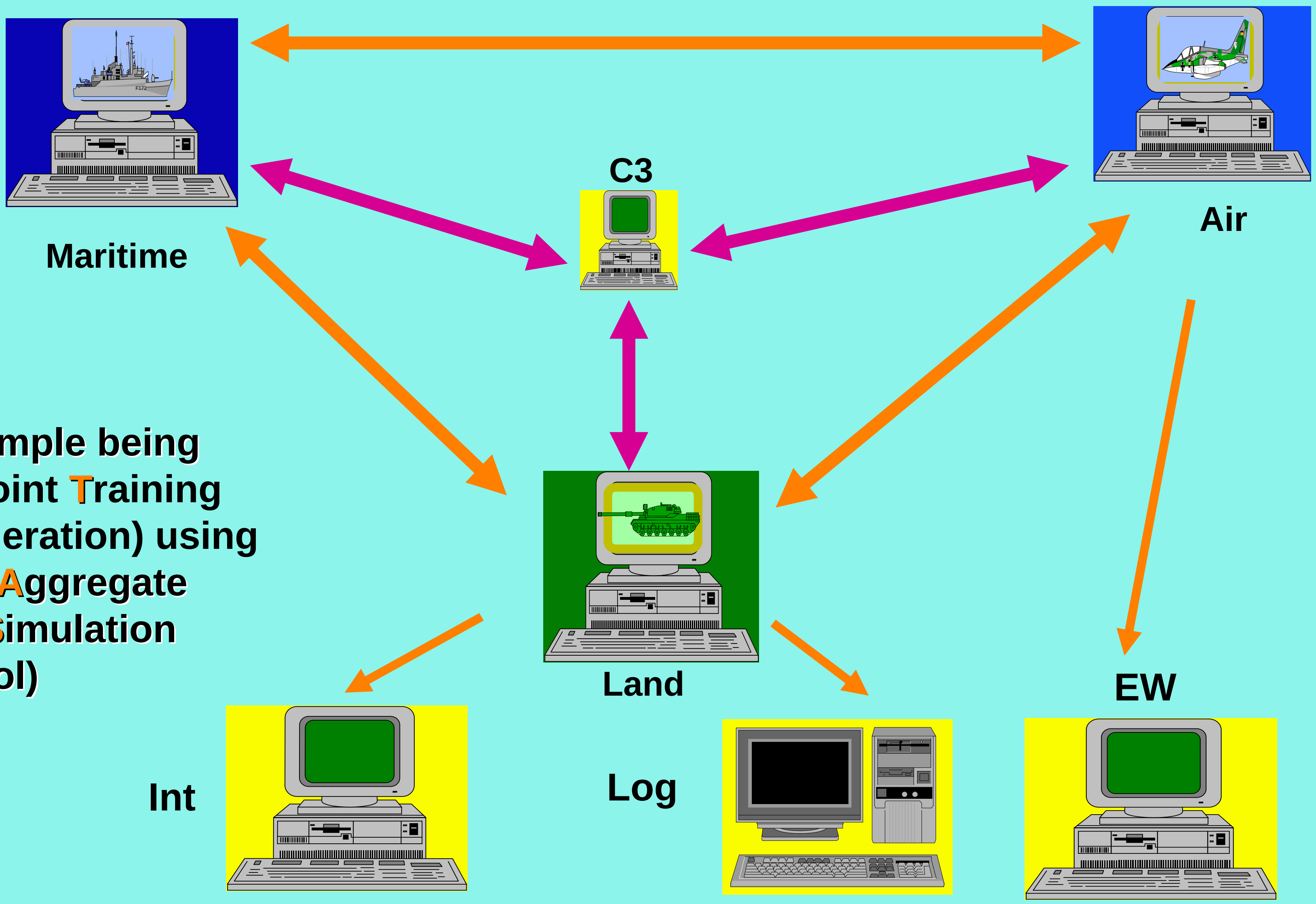
Matched Capabilities

[Overview of a Theory of Composability : Petty, Weisel & Mielke]

Distributed Simulations

- Networked federations of federates (M&S components)
- Different ways of 'distributing' (i.e. dividing up) the overall functionality needed, across multiple simulations on a network:
  - Distribute by <u>function</u> across federates:
    - Land battle, maritime battle, air battle components
    - Servers for weather, terrain, ballistics, lethality, vulnerability, sensors, motion, etc
  - Distribute by <u>platform</u> across federates:
    - Tank simulators, flight simulators, etc

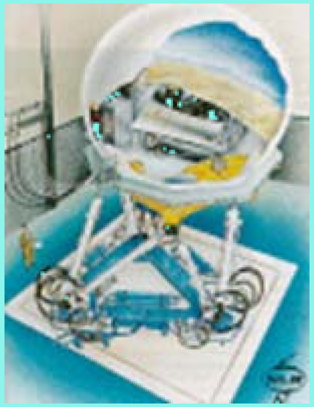
Distributed Simulation – By Function

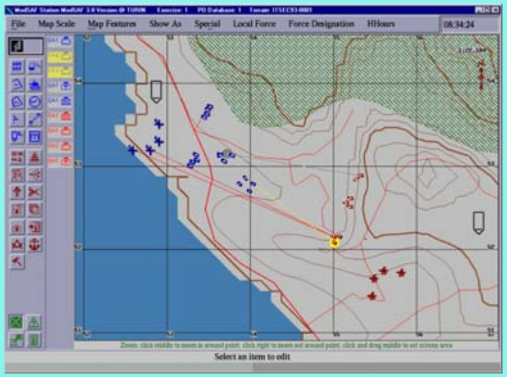


An example being JTC (Joint Training Confederation) using ALSP (Aggregate Level Simulation Protocol)

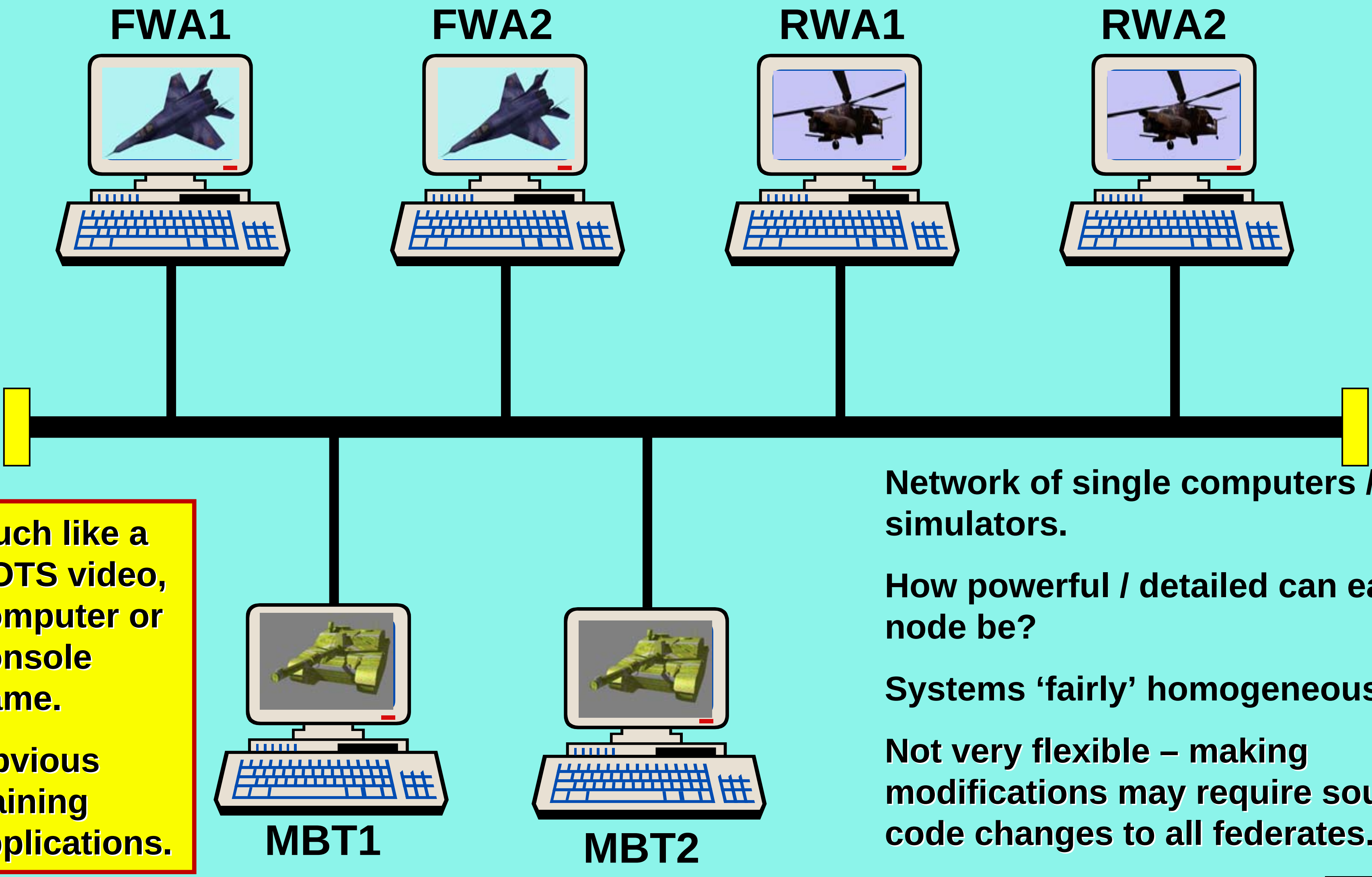
Distributed Simulation – By Platform

- IEEE 1278 DIS - Distributed Interactive Simulation
- Peer-to-Peer Distributed architecture of equals (i.e. of 'similar' components)
- Each station generates platform-level entities
- All sims/entities communicate with each other
- Messages (PDUs) about ground truth state are broadcast
- All stations potentially know all ground truth
- Many examples across all services and domains





Architectures for Distr.Sim. #1 – Discrete/Monolithic



Much like a COTS video, computer or console game.

Obvious training applications.

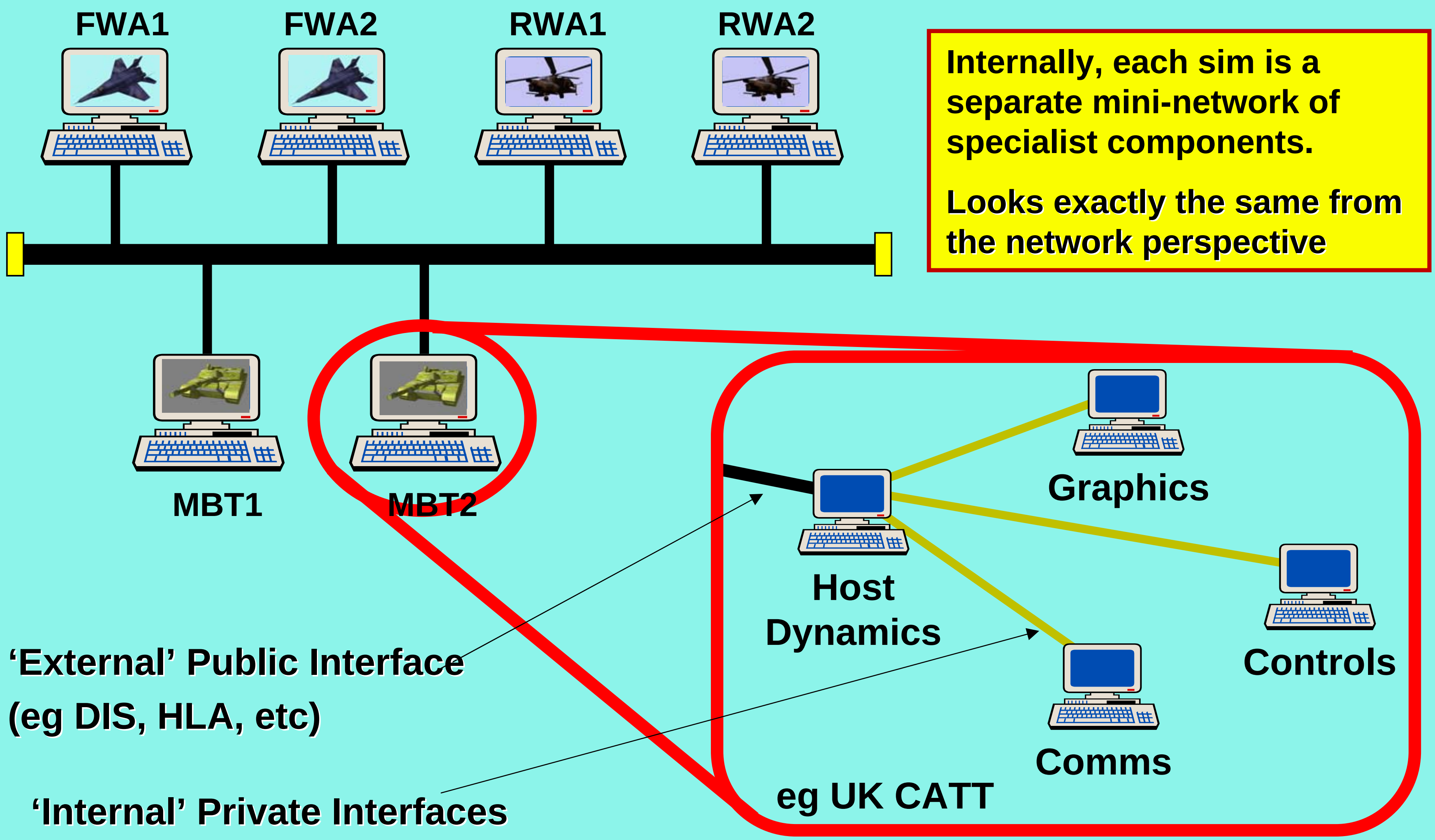
Network of single computers / simulators.

How powerful / detailed can each node be?

Systems 'fairly' homogeneous.

Not very flexible – making modifications may require source code changes to all federates.

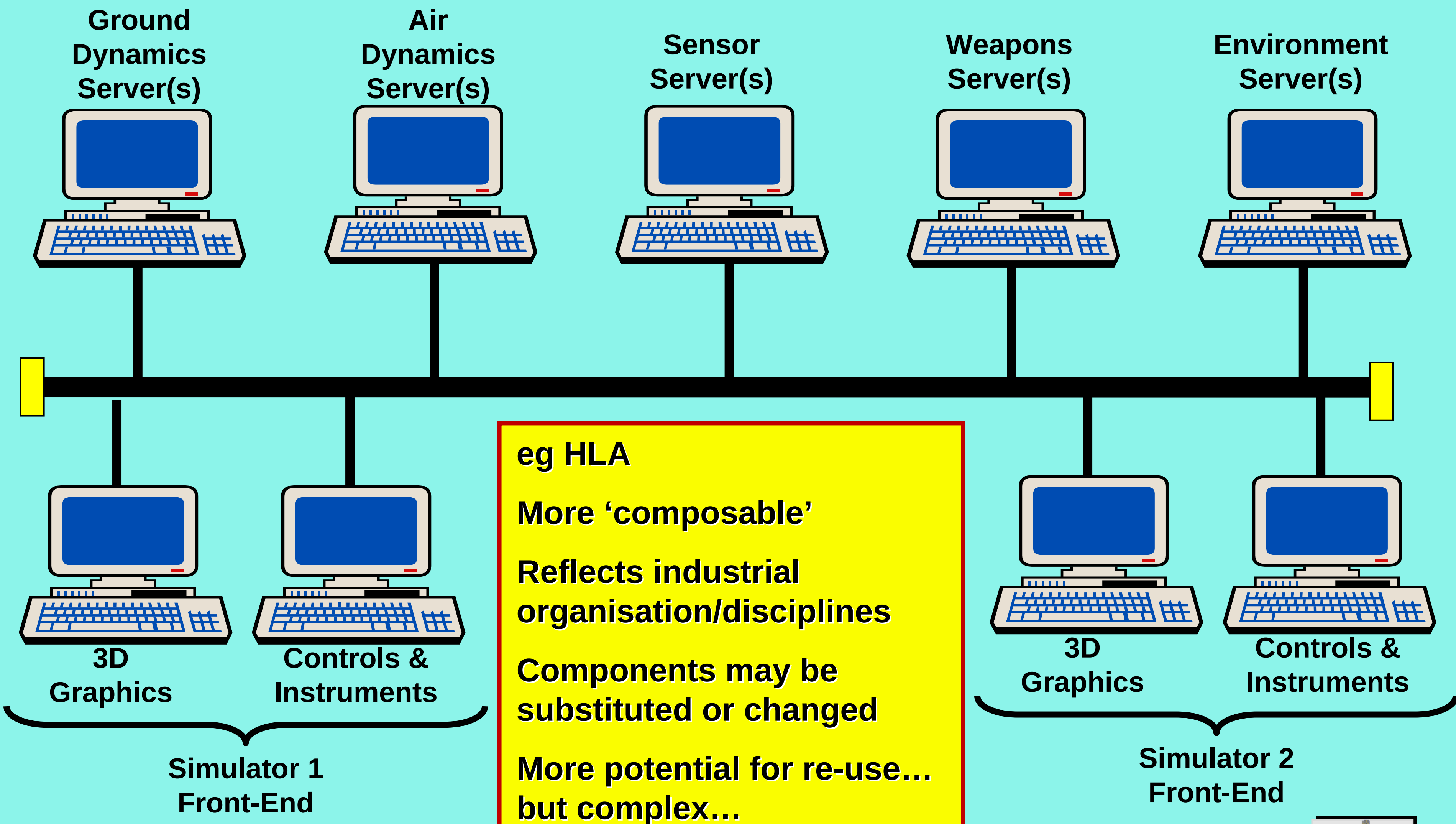
Architectures for Distr.Sim. #2 – Discrete/Component-based



Internally, each sim is a separate mini-network of specialist components.

Looks exactly the same from the network perspective

Architectures for Distr.Sim. #3 - Distributed Functional Components



eg HLA

More ‘composable’

Reflects industrial organisation/disciplines

Components may be substituted or changed

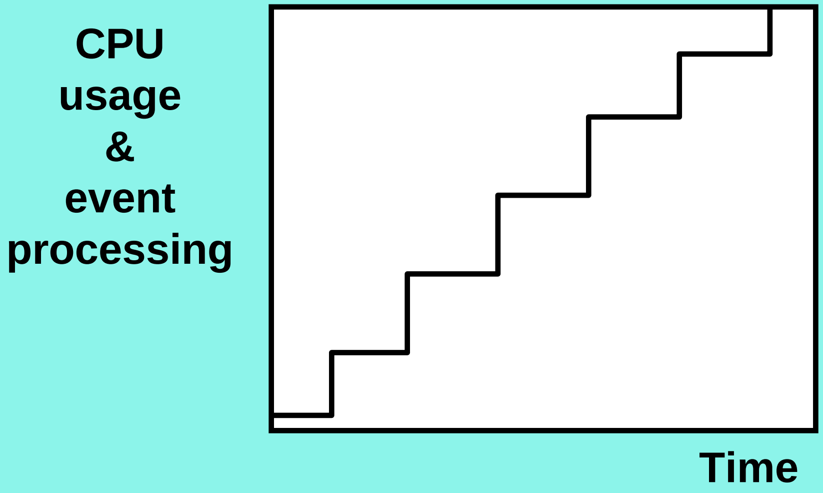
More potential for re-use… but complex…

Simulation Interoperability

- When looking to build a federation, it is clearly a good start point if all the multiple federates involved seem to represent the same sort of things in the same sort of way…
- So as an example, consider two hypothetical battlegroup-level, ground combat simulations which both represent individual vehicle platforms:
  - an older 'legacy' model, developed for stand-alone analysis use.
  - a newer model, developed from the outset as a training component.
- How interoperable might they be?
- Could the 'older' simulation be modified to make it interoperate with the newer one (ie to achieve re-use the old model) ?
- Even if it could, would it be <u>sensible</u> to do so? (NB 'legacy' vs 'pedigree'!)

Simulation Interoperability Example - 1

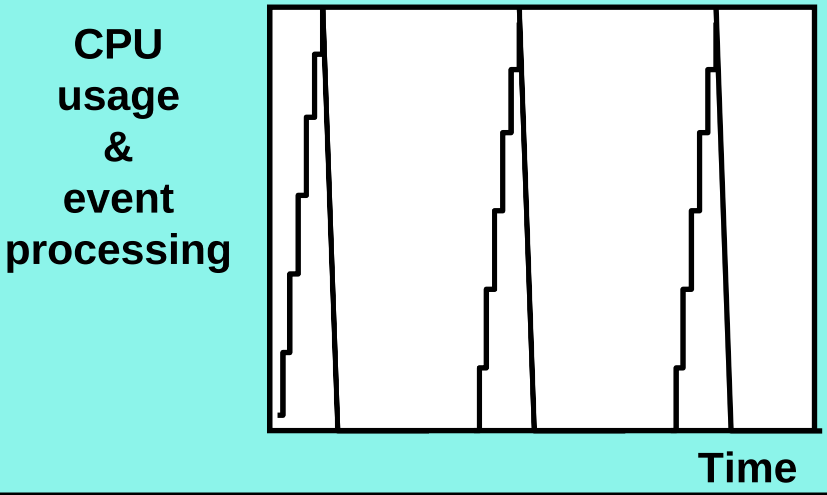
- Newer model:
  - Often 'Object oriented', modern coding languages/styles
  - Event-driven & callbacks Proper real-time software - usually cyclic, with very small time steps, consistent / smooth processing and time-advance
  - Well designed external interface(s).
  - Open system ('expects' to interact with remote/foreign entities)



Events occur (ie are generated, scheduled & processed) gradually over time – eg in 'real' time

Simulation Interoperability Example - 2

- Traditional combat model:
  - Older language & coding style, traditional (monolithic) coding.
  - Closed system (expects to own / control / update all entities).
  - Discrete event simulation, speed of execution is complexity dependant.
  - If running in real-time, executes small battle slice as quickly as possible (MUCH faster than real-time), then simply pauses (idle) to let real-time catch up. 'Bursty' processing & time advance.



Events generated and processed as fast as possible. If set to 'real-time' mode, simulation simply waits idle to let real-time catch up.

Simulation Interoperability Example - 3

- **Terrain:**

– **Old:**	**matrix of cells (height & features)**
– **New:**	**'often' polygon based (but doesn't have to be)**

- **Vehicle orientation:**

– **Old:**	**vehicle location & general orientation**
– **New:**	**vehicle x/y/z & h/p/r + turret, gun, etc orientation**

- **Vehicle space occupation:**

– **Old:**	**vehicles are abstract 'locations' only (no collisions)**
– **New:**	**vehicles occupy physical space & can collide.**

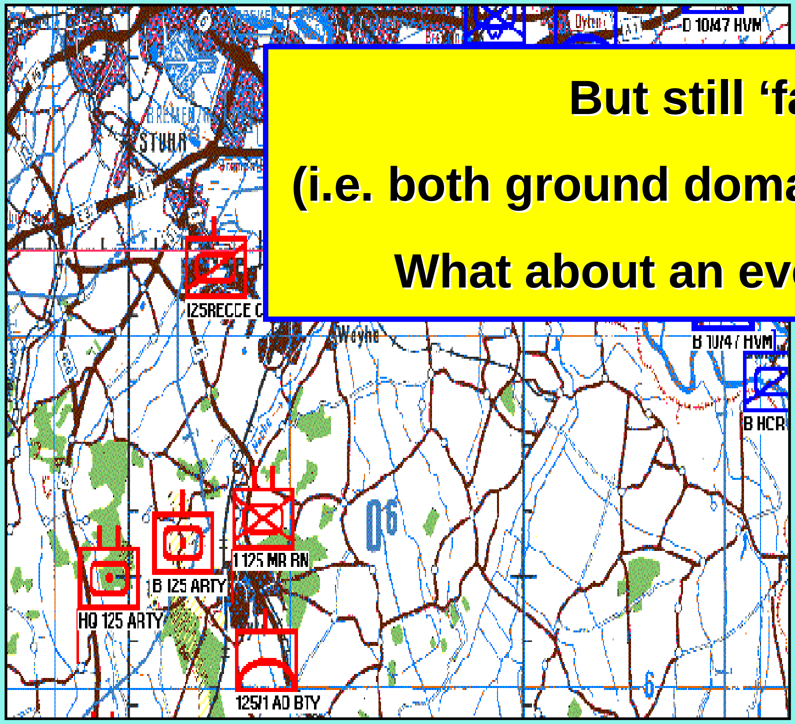
Simulation Interoperability Example - 4

- **Movement:**
  - **Old:** **known position, updated in discrete timeslices**
  - **New:** **known position; more continuously updated**
- **Engagement:**
  - **Old:** **PHit & PKill based. Model determines all effects.**
  - **New:** **Firer does ballistics & impact pt. Targets calc effects**
- **And remember, these are two models that are 'conceptually' quite similar to start with – i.e. both represent individual ground vehicle platforms!**
- **But on inspection, they represent that concept in rather different ways.**
- **Imagine therefore, how much more challenging it could be to interoperate models that are 'conceptually' even more different…..**

Simulation Interoperability

- So when considering Simulation Interoperability, we need to consider a range of issues…
- The initial problem is how to share simulation state data when hindered by the physics, characteristics and topology of networks. Even with modern networked systems, we should not ignore or underestimate the impact and costs of these issues.
- Issues such as security and IPR must also be addressed.
- But beyond that, it also concerns meaningful interpretation and use of the data thus exchanged by all the simulations / federates / tools involved.
  - i.e. we may be able to exchange data and even then understand it having done so, but does it actually make sense to do so?
- We must consider not just the implementation level, but the more conceptual levels underpinning it.

Linking More Heterogenous Simulations

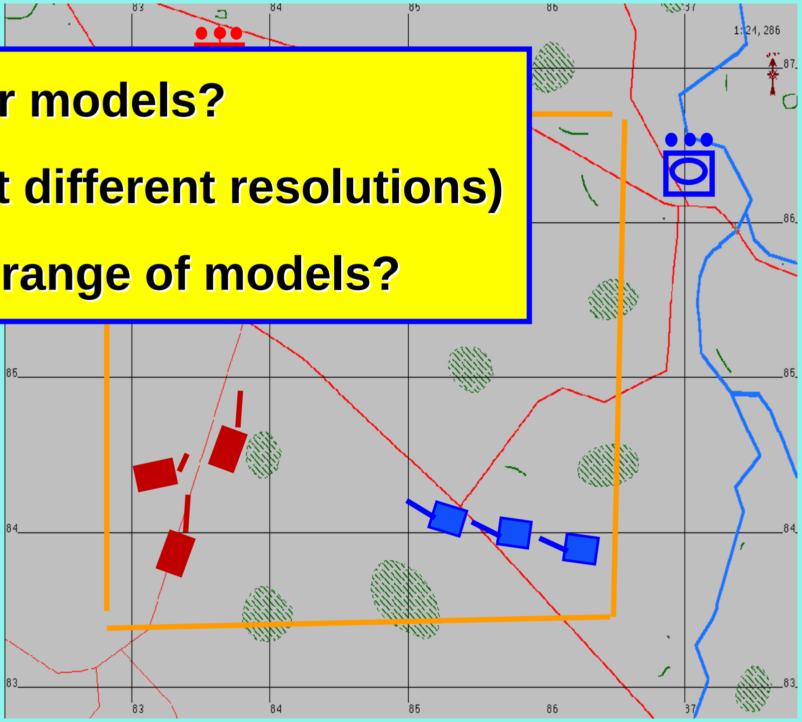


But still ‘fairly’ similar models?

(i.e. both ground domain, albeit at different resolutions)

What about an even broader range of models?

$\longrightarrow$? $\longleftarrow$



Aggregate Simulation

“101 Bn attacking ZX Regt”
“Firepower Score = 1502”
“Kill rate = 0.01”
“47% casualties”
“Advance Time Step + 15 mins”

Entity/Platform Simulation

“Tank 34B firing at tank 42A”
“SSKP = 92%”
“Mean time between shots = 5 secs”
“Object 42A now has an M-Kill”
“Time jump +0.04 secs to next event”

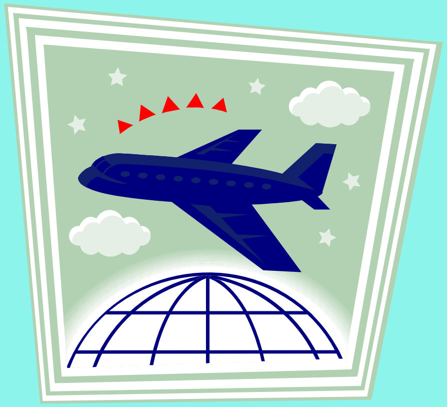
Joint Ops - Different Views of the World

Aggregated ground combat model

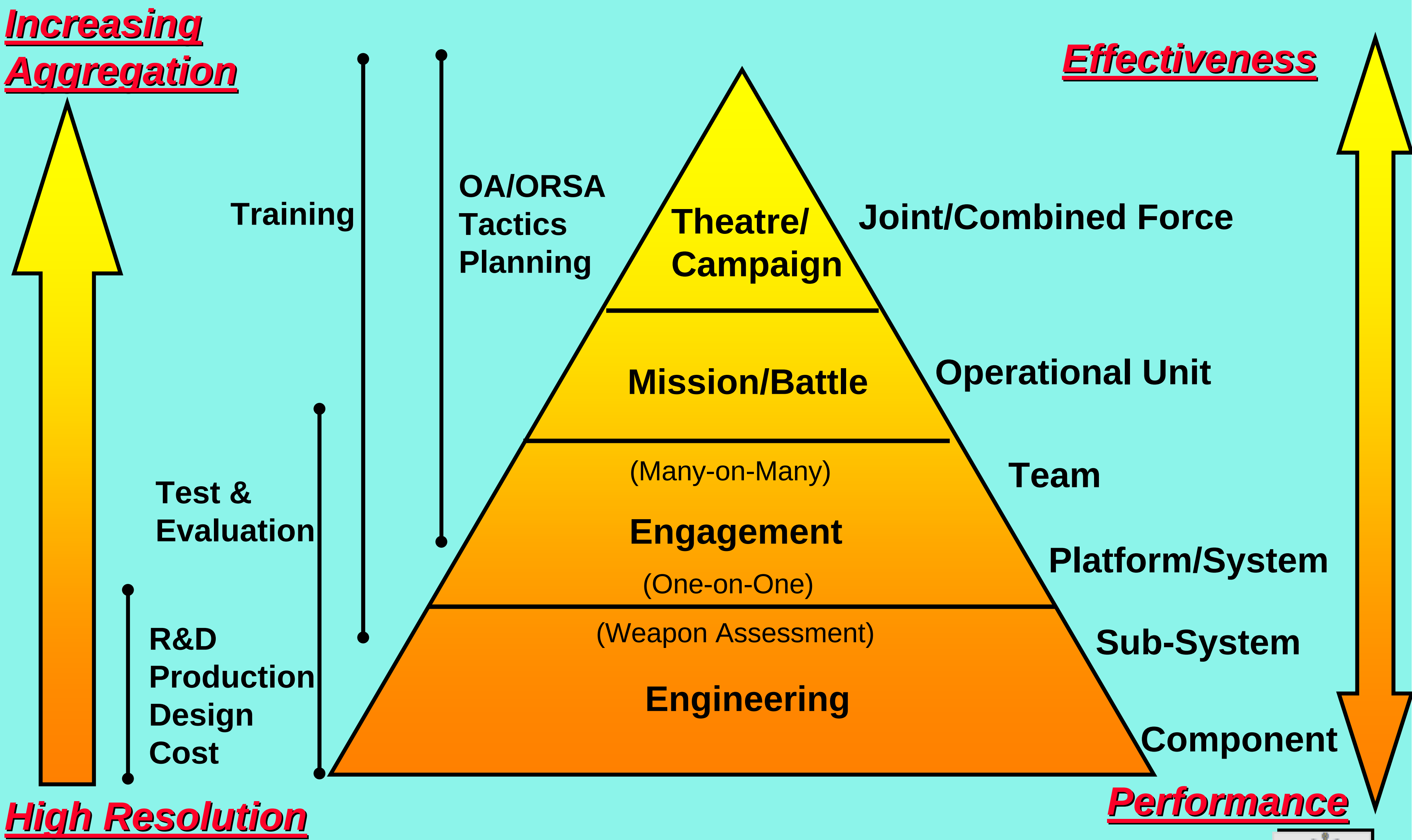
- Basic simulation object = Coy/Sqn to BG
- Flat earth database with 3D digital features in some detail.
- Uses UTM or MGRS coordinates.
- All entities are modelled the same (ie as 'vehicles'), some just have non-zero 'Z' coordinates

Aggregated air combat model

- Basic simulation object = air package
- Curved earth database with few / no 3D features
- Uses Lat-Long coordinates
- Simplified representation of ground combat as static or simply moving 'targets'.

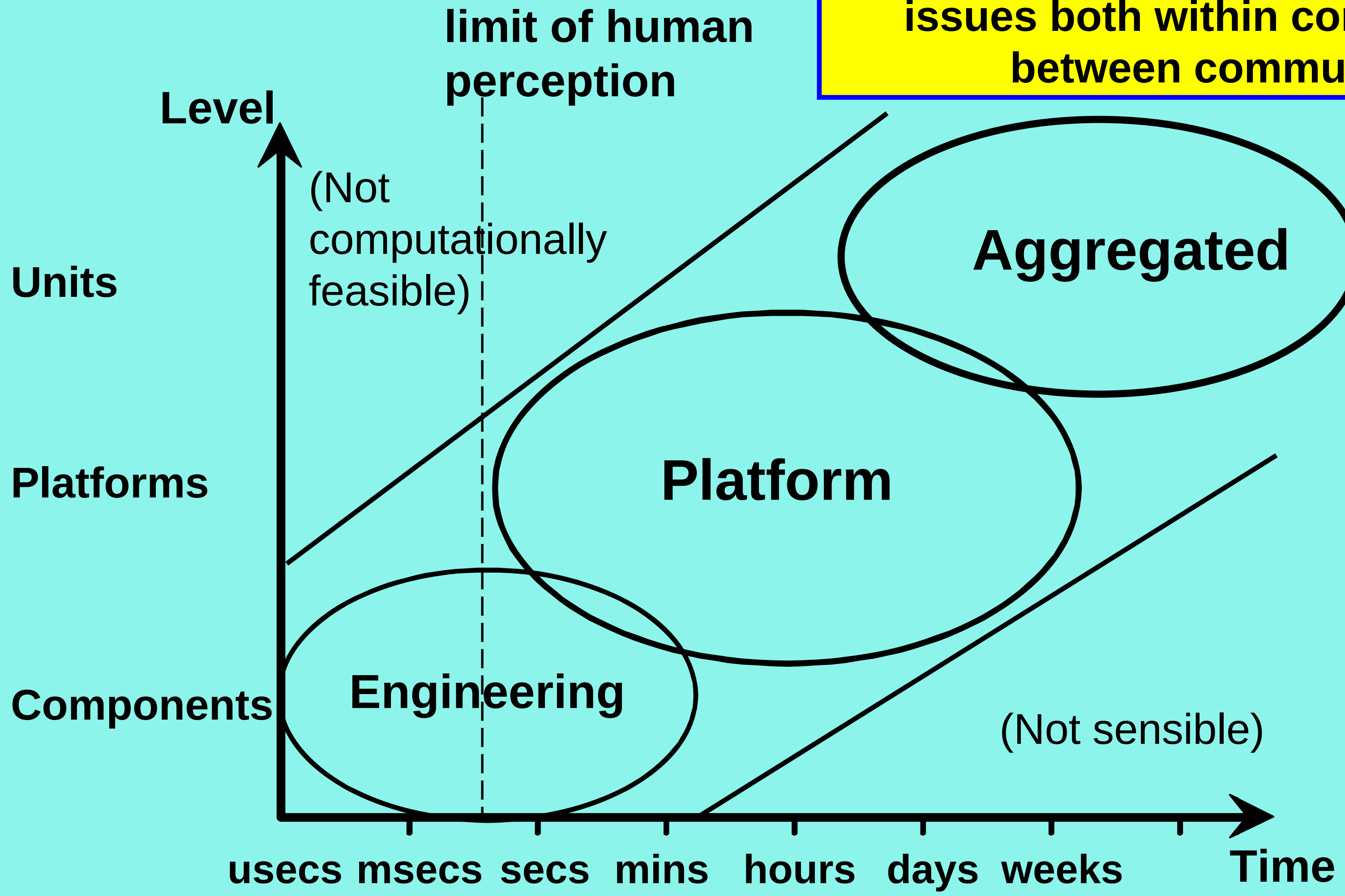


Hierarchy of Defence Modelling & Simulation



M&S Range of Applications

Evolution of like-minded '*Communities of Interest*' – hence interoperability issues both within community and between communities…



LCIM - Levels of Conceptual Interoperability Model

- **Level 0 - No connection**
- **Level 1 - Technical Level**

  - Physical connectivity (bits and bytes).
- **Level 2 - Syntactical Level**

  - Standardized data formats (i.e. protocols).
- **Level 3 - Semantic Level**

  - Data & context (i.e. information)

  - Common reference models for unambiguous meaning of data.
- **Level 4 - Pragmatic/Dynamical level**

  - Information and its use and application (i.e. knowledge)
- **Level 5 - Conceptual Level**

  - Common world view

  - Common view of interrelations between elements

[Tolk & Muguira, 2003 Fall SIW & Tolk 2004, Spring SIW]

The Evolution of Distributed Simulation

Increasing scope, scale range and type of both networked simulations and their applications (Trg, R&D, etc).

- SIMNET (US) : Development	1983+
- SIMNET (US) : 'Fielded' Trainer	1991+
- AGPT (GER)	1991+
- DIS Development begun	1991
- DIS 1.0 (IEEE 1278.1993)	1993
- DIS 2.0 (IEEE 1278.1995)	1995
- DIS 2.1 (IEEE 1278.1998)	1998
- CCTT, CATT (US)	1999+
- CATT (UK)	2002+
- DIS ?.? (IEEE 1278.????)	2005+
- High Level Architecture	1995+
- ????	

Simnet: Platforms, Real-time, Homogenous, Proprietary.

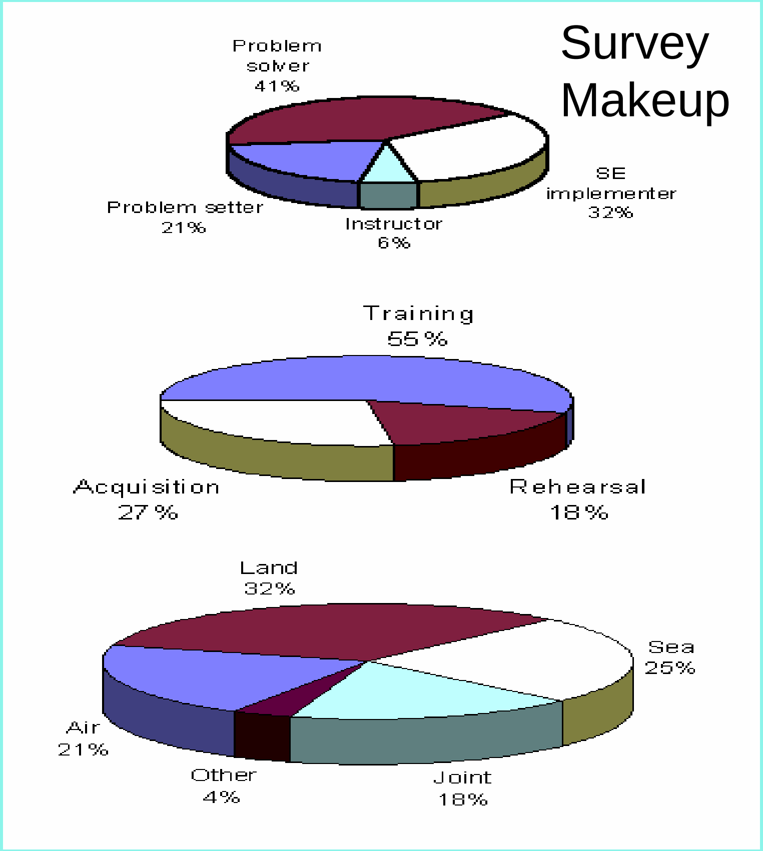
DIS: Platforms, Real-time, Heterogenous, Open.

HLA: Various levels, Various times, Heterogenous, Open.

◘ Resolution:	Sub-system, platform, aggregate units.
◘ Time-base:	Real-time, faster, slower, discrete-event.
◘ Computing:	Similar systems (homogenous) or not (heterogenous)
◘ Standard:	Open or Commercial/Proprietary

EUCLID RTP 11.13

- EUCLID definition: “An SE is considered to be the instantiation of a heterogeneous networked simulation”.
- One element of the programme specifically surveyed obstacles to the exploitation of Synthetic Environments in Europe.
- 9 categories of ‘obstacle’ identified:
  - People And Resources
  - Security
  - Cost
  - User Requirements
  - Management Issues
  - Availability Of Standards
  - Legacy Systems
  - Networks
  - Technical Issues



[“Realising the potential of Networked Simulation in Europe”, Dumetz & Fabri, SMI Virtual Battlefield Conference 2002]

EUCLID RTP 11.13 – Obstacles to SE's - 1

- **People And Resources**
  - **Customer education / awareness; Military staff turnover; Small SE community.**
- **Security**
  - **Different national & organisational rules; IPR implications; Protection of data.**
- **Cost**
  - **Return-on-Investment not proven; Time, resource & cost intensive.**
- **User Requirements**
  - **User-to-Developer interface issues; Mission creep during programmes.**
- **Management Issues**
  - **Heterogeneous team management difficult; Long legal / contractual processes.**

EUCLID RTP 11.13 – Obstacles to SE's - 2

- **Availability Of Standards**
  - Inadequate and/or inconsistent standards for VV&A and SNE representations.
- **Legacy Systems**
  - Adapting & re-using databases difficult; Documentation?; Migration costly.
- **Networks**
  - Network & RTI are both limitations; Long distance latencies; Network failures.
- **Technical Issues**
  - Speech input & output immature; Foolproof initialisation difficult.

Many of these obstacles are 'Integration' challenges, not 'Interoperability' issues.

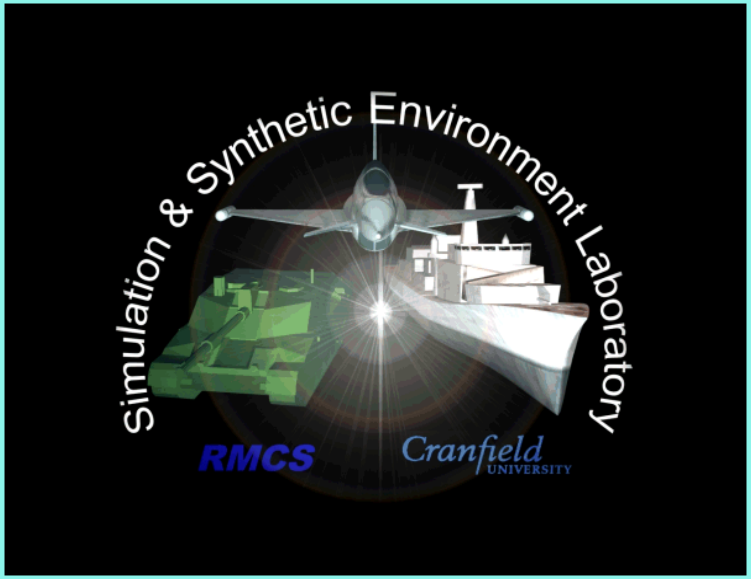
Summary

- Composability offers the potential for adaptation, flexibility and re-use.
- There are many ways, at many levels, of composing simulation solutions.
- Composability requires Interoperability, which must however be considered at many levels, not just the technical and engineering but also the conceptual.
- There are still many obstacles to be overcome in order for Synthetic Environments of federated and distributed simulations to gain widespread use.
- Not all of these challenges are 'technical/conceptual' ones to do with Modelling and Simulation Interoperability.

Questions / Discussion

Mr. Jonathan Searle
Manager, SSEL
Defence Academy
Cranfield University
+44 (0)1793 785852
j.r.searle@cranfield.ac.uk

Mr. John Brennan
Lecturer, Defence Simulation
Defence Academy
Cranfield University
+44 (0)1793 785464
j.m.brennan@cranfield.ac.uk



http://www.rmcs.cranfield.ac.uk/ssel